Département EEA - Faculté Sciences et Ingénierie

Master EEA SME

Compte rendu BE

Année 2022-2023

TP de base DHT22

Rédigé par : JIDAN Bilal

14 mai 2023

Table des matières

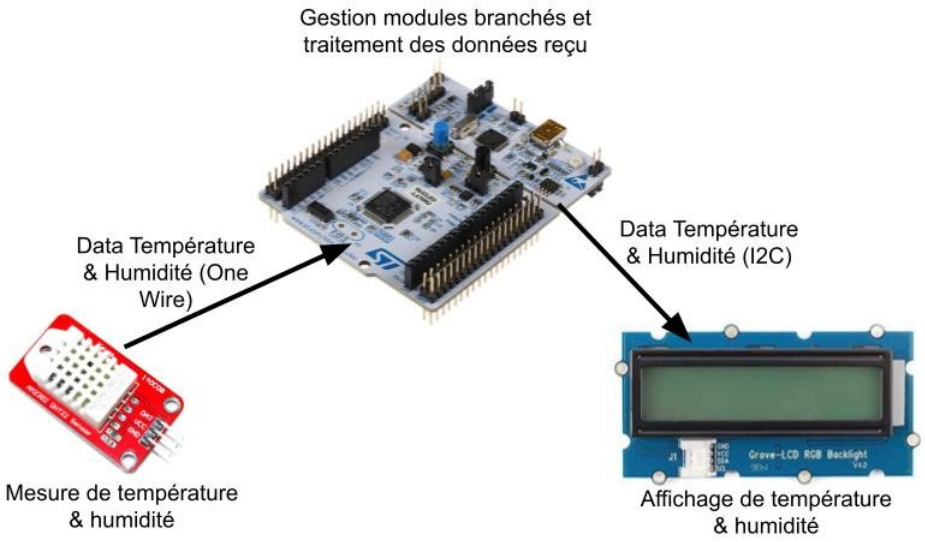
1 Introduction

L'objectif de ce projet est de créer un système qui peut mesurer avec précision la température et l'humidité de l'environnement et les afficher en temps réel sur un écran LCD. Cela peut être utile dans de nombreuses applications, telles que la surveillance climatique dans une maison, le contrôle de l'humidité dans une serre ou la gestion de l'environnement dans des laboratoires.
Au cours de ce projet, je vais explorer les différentes étapes nécessaires pour réaliser cette application.nous étudierons la programmation du microcontrôleur STM32 pour interagir avec le capteur et récupérer les données de température et d'humidité en utilisant le capteur DHT22 et un écran LCD.

2 Matériel/logiciel

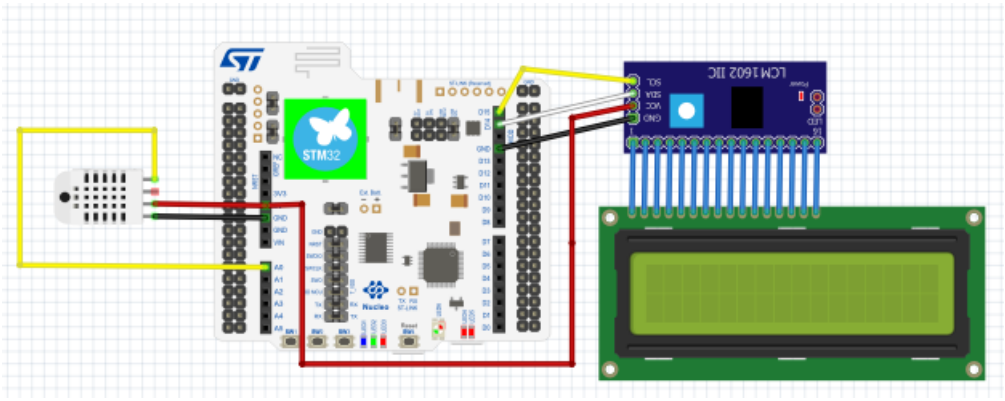
Matériel : Dans le cadre de ce projet, nous utilisons du matériel essentiel pour nos expérimentations. Cela comprend un microcontrôleur STM32L476rg, un écran LCD de type White on Blue, ainsi qu'un capteur de température et d'humidité DHT22.
Logiciel : Nous utilisons le logiciel CubeIDE, qui est spécialement conçu pour la configuration et la programmation du microcontrôleur STM32L476rg. Il nous offre un environnement de développement intégré, permettant de simplifier la programmation et la gestion des paramètres de notre microcontrôleur.

3 Schéma fonctionnel



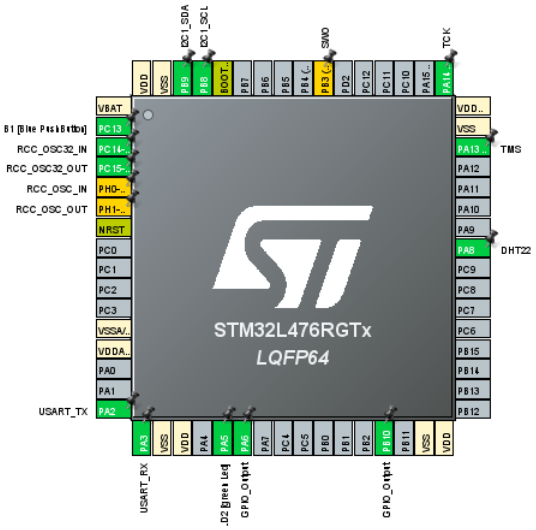
Dans ce schéma, le microcontrôleur STM32L476rg communique avec le capteur DHT22 via le protocole I2C. Le microcontrôleur envoie une demande de mesure au capteur, qui répond ensuite avec les données de température et d'humidité. Une fois que le microcontrôleur a récupéré les données, il les envoie à l'écran LCD pour les afficher. L'écran LCD est utilisé pour visualiser en temps réel les informations de température et d'humidité fournies par le capteur DHT22. Cette communication bidirectionnelle permet au microcontrôleur de contrôler le capteur DHT22 et de recevoir les données mesurées, puis de les afficher sur l'écran LCD pour une visualisation pratique et conviviale.

4 Schéma de câblage (fritzing)



5 Initialisation du microcontrôleur STM32L476rg

Avant toute chose, il faut d'abord paramétrer notre microcontrôleur selon notre utilisation.



Pour commencer, nous devons configurer la communication avec l'écran LCD, qui utilise le protocole I2C. Sur le microcontrôleur, nous devons définir les ports SDA et SCL pour

établir la communication avec l'écran LCD. Dans le schéma précédent, nous pouvons voir que les ports utilisés pour la communication sont PB8 et PB9.

En ce qui concerne le capteur de température et d'humidité DHT22, il utilise un protocole de communication appelé One Wire. Pour permettre cette communication, nous devons attribuer une broche du microcontrôleur au capteur. Sur le schéma ci-dessus, nous pouvons observer que la broche utilisée est PA8. Selon la documentation technique (datasheet) du capteur, cette broche doit être maintenue à l'état logique 1 lorsqu'elle est au repos. Par conséquent, nous la configurerons en tant que pull-up.

Enfin, il est important de configurer un timer en microsecondes, car cela est nécessaire pour assurer le bon fonctionnement du capteur DHT22.

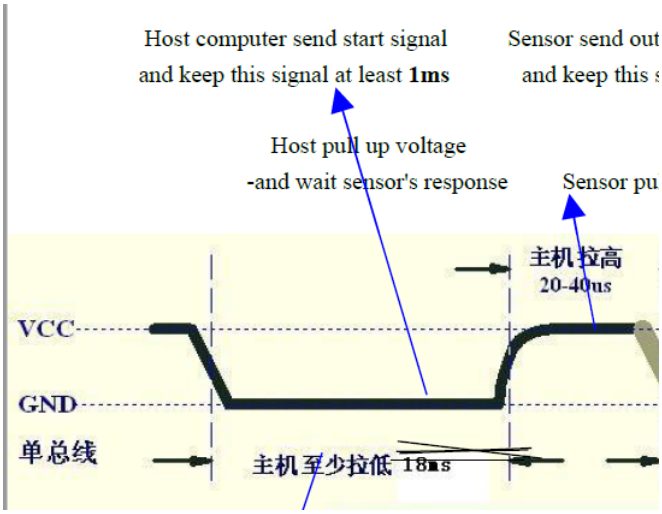
Ainsi, en prenant en compte ces différentes configurations, nous pourrons établir une communication correcte entre le microcontrôleur STM32 et à la fois l'écran LCD et le capteur DHT22.

6 Initialisation du capteur DHT22

Nous allons commencer par créer une fonction pour notre temporisation en microsecondes, appelée "Delay_us".

```c
void Delay_us(uint16_t us)
{
    __HAL_TIM_SET_COUNTER(&htim2, 0);
    while(__HAL_TIM_GET_COUNTER(&htim2) < us);
}
```

Ensuite, nous devons consulter la datasheet du capteur DHT22 pour comprendre comment il s'initialise.

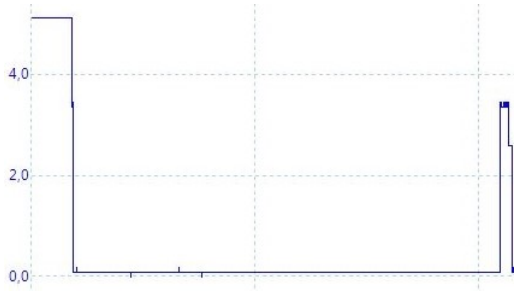


D'après les informations de la datasheet du capteur DHT22, nous devons effectuer une séquence d'initialisation spécifique pour établir une communication réussie. Voici la procédure d'initialisation reformulée :

1 : Mettre la broche de signal reliant le microcontrôleur et le DHT22 à l'état logique bas (0) pendant au moins 1 milliseconde (ms) et au maximum 18 ms. Cela permet d'indiquer au capteur que le microcontrôleur est prêt à communiquer.

2 : Passé le délai d'attente initial, mettre la broche de signal à l'état logique haut (1) pendant une durée de 20 à 40 microsecondes (µs). Cette impulsion indique au capteur que

le microcontrôleur est prêt à recevoir des données.



Lorsque nous comparons le graphe obtenu à l'aide du picoscope avec celui présent dans la datasheet, nous pouvons observer que notre séquence d'initialisation est en accord avec les spécifications.

Tout d'abord, nous constatons que notre broche de signal est à l'état logique 1 au repos, ce qui correspond à la spécification de la datasheet. Ensuite, nous effectuons une transition à l'état logique 0 et maintenons cette valeur pendant exactement 1,2 millisecondes (ms), ce qui correspond également à la durée spécifiée dans la datasheet.

Cependant, nous remarquons une légère variation de tension à droite du graphique, au moment où nous souhaitons revenir à l'état logique 0. Cette petite variation de hauteur est causée par le changement de direction du PIN, lorsque nous le passons en mode entrée. Cela peut entraîner une transition progressive de la tension avant qu'elle n'atteigne finalement l'état logique 0.

Il est important de noter que cette variation de tension n'affecte généralement pas de manière significative la communication avec le capteur, tant que nous respectons les délais et les niveaux logiques spécifiés dans la séquence d'initialisation.

```
Data_Output(GPIOA, GPIO_PIN_8); //info vers le capteur
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
DWT_Delay_us(1200); //signal de commande
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
DWT_Delay_us(30); //signal de commande
Data_Input(GPIOA, GPIO_PIN_8); //info vers le microcontroleur

/*commence la reception de donnees*/

while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8)));

for (k=0;k<1000;k++)
{
    if (HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_8) == GPIO_PIN_RESET)
    {
      break;
    }
}

while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8)));
DWT_Delay_us(40);
```

1 : Data_Output(GPIOA, GPIO_PIN_8) : Cette fonction configure la broche GPIOA_PIN_8 en tant que sortie, permettant ainsi d'envoyer des informations au capteur.

2 : HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET) : Cette instruction met la broche GPIOA_PIN_8 à l'état logique bas (0). Cela indique au capteur que le microcontrôleur est prêt à établir une communication.

3 : DWT_Delay_us(1200) : Cette fonction introduit un délai de 1200 microsecondes (1,2 millisecondes). Ce délai correspond à la spécification de la datasheet, où la broche doit être maintenue à l'état logique bas pendant 1,2 ms avant de passer à l'état logique haut.

4 : HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET) : Cette instruction met la broche GPIOA_PIN_8 à l'état logique haut (1). Cela indique au capteur que le microcontrôleur est prêt à recevoir des données.

5 : DWT_Delay_us(30) ; : Cette fonction introduit un délai de 30 microsecondes. Ce délai correspond à la spécification de la datasheet, où la broche doit être maintenue à l'état logique haut pendant 20 à 40 microsecondes.

6 : Data_Input(GPIOA, GPIO_PIN_8) : Cette fonction configure la broche GPIOA_PIN_8 en tant qu'entrée, permettant ainsi au microcontrôleur de recevoir les informations provenant du capteur.

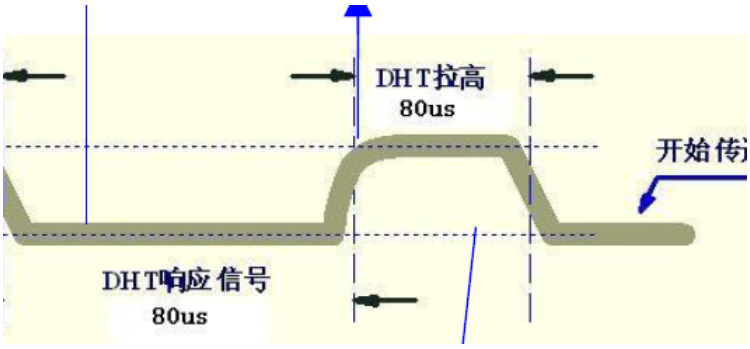
7 : while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8))) : Cette boucle attend que la broche GPIOA_PIN_8 passe à l'état logique haut, indiquant que le capteur est prêt à transmettre des données.

8 : for (k=0 ;k<1000 ;k++) : Cette boucle vérifie si la broche GPIOA_PIN_8 reste à l'état logique haut. Si elle passe à l'état logique bas, la boucle est interrompue.

9 : while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8))) : Cette boucle attend que la broche GPIOA_PIN_8 revienne à l'état logique haut après avoir été à l'état logique bas. Cela indique que le capteur a terminé la transmission de données.

10 : DWT_Delay_us(40) : Cette fonction introduit un délai de 40 microsecondes. Ce délai correspond à une période d'attente après la fin de la transmission de données du capteur.

7 Réponse du DHT22



Si l'initialisation du capteur DHT22 est réussie, il répondra au microcontrôleur en envoyant un signal spécifique. Ce signal commence par une impulsion à l'état logique bas (0) qui dure environ 80 microsecondes (µs), suivie d'une impulsion à l'état logique haut (1) également d'une durée d'environ 80 µs. Ensuite, il y a une autre impulsion à l'état logique bas qui dure environ 50 µs. Après cette séquence, le capteur envoie un bit de donnée qui est

à l'état logique haut (1) et marque le début de la transmission des données proprement dite.

8 Des calculs pour obtenir les valeurs d'humidité et de température

```
Read_data(&dataH1);
Read_data(&dataH2);
Read_data(&dataT1);
Read_data(&dataT2);
Read_data(&SUM);

check = dataH1 + dataH2 + dataT1 + dataT2;

RH = (dataH1<<8) | dataH2;
TEMP = (dataT1<<8) | dataT2;

Humidite = RH / 10.0;
Temperature = TEMP / 10.0;
```

1 : Read_data(&dataH1) ; : Cette ligne de code appelle une fonction Read_data() pour lire le premier octet de données de l'humidité (dataH1) envoyé par le capteur DHT22. L'opérateur & est utilisé pour passer l'adresse de la variable dataH1 afin que la fonction puisse y stocker la valeur lue.

2 : Les lignes suivantes (Read_data(&dataH2) ;, Read_data(&dataT1), Read_data(dataT2) ;, Read_data(&SUM) ;) effectuent des opérations similaires pour lire les autres octets de données (dataH2, dataT1, dataT2) et la somme de contrôle (SUM) envoyés par le capteur DHT22.

3 : check = dataH1 + dataH2 + dataT1 + dataT2 : Cette ligne de code calcule la somme des octets de données (dataH1, dataH2, dataT1, dataT2) pour obtenir une valeur de vérification (check). Cette somme peut être utilisée pour vérifier l'intégrité des données lues. Il peut être comparé à d'autres calculs ou à une valeur attendue pour détecter les erreurs de lecture.

4 : RH = (dataH1<8) | dataH2 ; : Cette ligne de code combine les deux octets de données de l'humidité (dataH1 et dataH2) pour obtenir une valeur entière de l'humidité relative (RH). L'opérateur de décalage vers la gauche (<) est utilisé pour déplacer les bits de dataH1 de 8 positions vers la gauche, puis l'opérateur de "ou" binaire (|) combine les bits décalés avec ceux de dataH2 pour obtenir la valeur entière de l'humidité relative.

5 : TEMP = (dataT1«8) | dataT2 : Cette ligne de code combine les deux octets de données de la température (dataT1 et dataT2) de manière similaire à l'étape précédente, pour obtenir une valeur entière de la température (TEMP).

6 : Les lignes suivantes (Humidite = RH / 10.0, Temperature = TEMP / 10.0)s convertissent les valeurs de l'humidité et de la température en utilisant une division par 10.0. Cela permet d'obtenir les valeurs réelles de l'humidité et de la température avec une précision décimale.

9 Affichage des données obtenu sur l'écran LCD

```
sprintf(bufRH,"Humidite: %.1f", Humidite);
sprintf(bufT, "Temp.: %.1f C", Temperature);
lcd_position(&hi2c1,0,0);
lcd_print(&hi2c1,bufRH);
lcd_print(&hi2c1,"%");
lcd_position(&hi2c1,0,1);
lcd_print(&hi2c1,bufT);
reglagecouleur(0,0,255);
```

Les lignes de code que vous avez fournies sont utilisées pour formater les valeurs d'humidité et de température et les afficher sur un écran LCD.

pour plus de detaille :

1 : sprintf(bufRH,"Humidite : %.1f", Humidite) : Cette ligne de code utilise la fonction sprintf pour formater la valeur d'humidité (Humidite) avec une précision décimale de 1 chiffre après la virgule. La valeur formatée est stockée dans la variable bufRH, qui est une chaîne de caractères. La chaîne de format "Humidite : %.1f" indique que la valeur de l'humidité sera insérée à la place de "%.1f" dans la chaîne.

2 : sprintf(bufT, "Temp. : %.1f C", Temperature) : Cette ligne de code fait la même chose que la précédente, mais cette foisci pour la valeur de température (Temperature). La chaîne de format "Temp. : %.1f C" indique que la valeur de la température sera insérée à la place de "%.1f" dans la chaîne, suivie de l'unité de mesure "C" pour Celsius.

3 : lcd_position(&hi2c1,0,0) : Cette ligne de code positionne le curseur de l'écran LCD à la première ligne (ligne 0) et à la première colonne (colonne 0).

4 : lcd_print(&hi2c1,bufRH) : Cette ligne de code affiche la valeur formatée de l'humidité (bufRH) sur l'écran LCD. La fonction lcd_print() est utilisée pour cela.

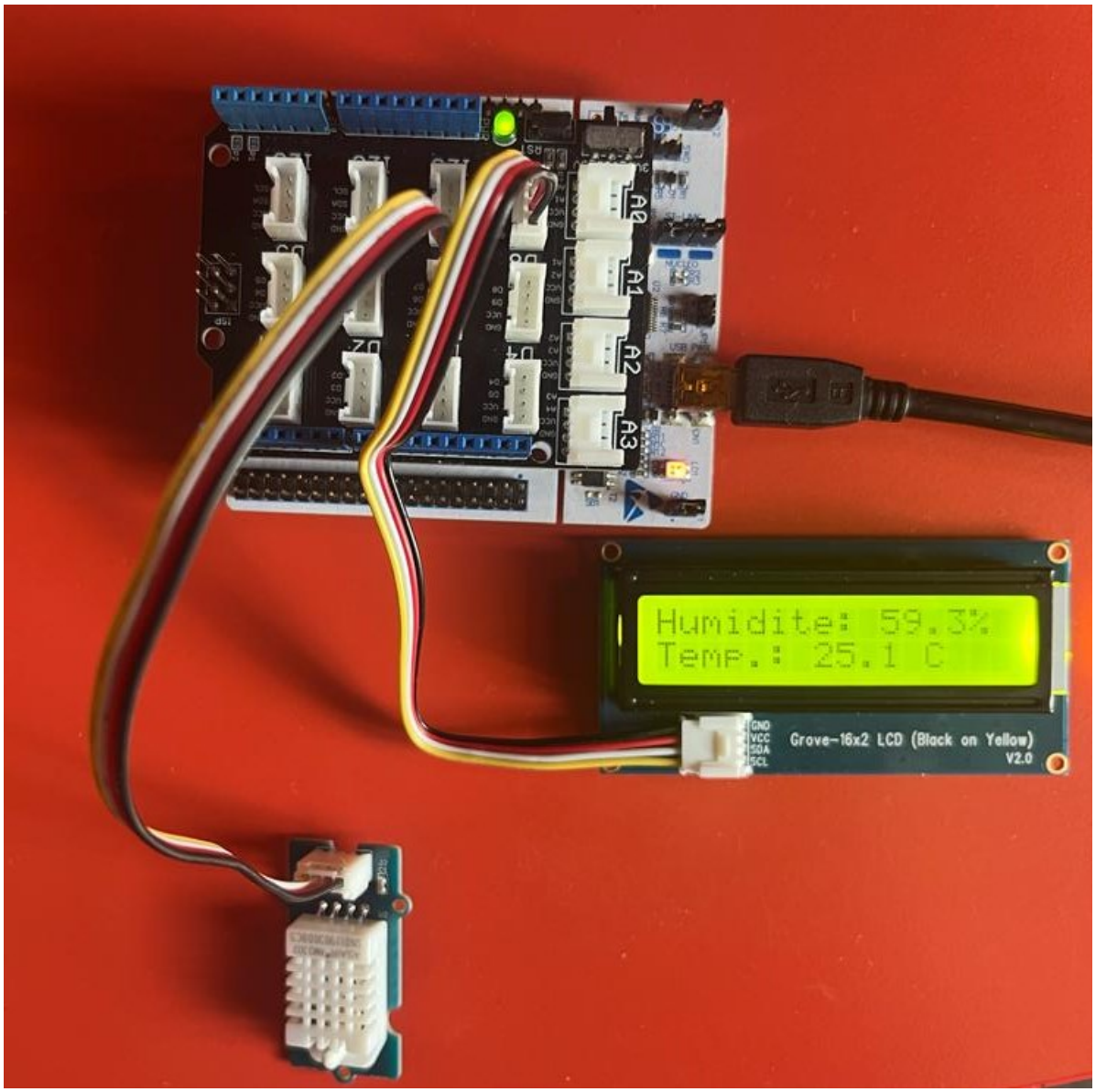
5 : lcd_print(&hi2c1,"%") ; : Cette ligne de code affiche le symbole "%" sur l'écran LCD. Il est utilisé pour indiquer l'unité de mesure de l'humidité.

6 : lcd_position(&hi2c1,0,1) ; : Cette ligne de code positionne le curseur de l'écran LCD à la deuxième ligne (ligne 1) et à la première colonne (colonne 0).

7 : lcd_print(&hi2c1,bufT) ; : Cette ligne de code affiche la valeur formatée de la température (bufT) sur l'écran LCD.

10 Resultat final de projet de bases

Voici ci-dessous une photo du résultat final du projet, affichant les valeurs d'humidité et de température sur l'écran LCD :


A0
A1
A2
A3
Humidite: 59.3%
Temp.: 25.1 C
GND
VCC
SDA
SCL
Grove-16x2 LCD (Black on Yellow)
V2.0

11 Conclusion

En conclusion, ce projet basé sur le microcontrôleur STM32L476rg, le capteur de température et d'humidité DHT22 et l'écran LCD a été réalisé avec succès. J'ai configuré les ports du microcontrôleur pour la communication I2C avec l'écran LCD et attribué une broche pour la communication One Wire avec le capteur DHT22. J'ai également mis en place une fonction de temporisation en microsecondes pour assurer le bon fonctionnement du capteur.

En utilisant les spécifications fournies par la datasheet du DHT22, nous avons réussi à initialiser le capteur et à obtenir les données d'humidité et de température. Les valeurs lues ont été vérifiées à l'aide de la somme des octets de données, et j'ai combiné les octets pour obtenir les valeurs finales d'humidité et de température.

Ce projet démontre la capacité du microcontrôleur STM32L476rg à interagir avec des capteurs et à afficher les données sur un écran LCD. Il peut servir de base pour des applications plus avancées impliquant la collecte de données environnementales telles que la surveillance de la température et de l'humidité dans divers contextes.